UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/842,270	04/24/2001	Michael J. Grier	2501	5601

7590 10/04/2006

Law Offices of Albert S. Michalik, PLLC
704-228th Avenue NE
Suite 193
Sammamish, WA 98074

EXAMINER
YIGDALL, MICHAEL J

ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 10/04/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/842,270	GRIER ET AL.	
	Examiner	Art Unit	
	Michael J. Yigdall	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12 June 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-49* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-49* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on June 12, 2006 has been entered. Claims 1-49 are pending.

Response to Arguments

2. Applicant's arguments with respect to Saboff have been considered but are moot in view of the new ground(s) of rejection, as set forth below. Applicant's amendment necessitated the new ground(s) of rejection.

3. Applicant's arguments with respect to Hammond have been fully considered but they are not persuasive.

Applicant contends that Hammond leaves unaddressed any scenario in which two versions of the same module may be located in the same directory (remarks, page 21, top), and similarly that Hammond cannot handle a scenario wherein versions of the same module are located in the same directory structure (remarks, page 29, bottom).

Notwithstanding Applicant's characterization of Hammond, there is no basis for this argument in the claims. The examiner recognizes that Applicant's argument is that the present invention distinguishes among different versions of the same assembly located in the same directory. However, this is narrower than what is now recited in the claims. For example, claim

1, as amended, recites that the assembly is "among a plurality of assemblies having at least some components located in a same directory" and further recites an "activation context that distinguishes between versions of assemblies." The language of the claim does not call for different versions of the *same* assembly located in the same directory. Instead, the claim merely indicates that several assemblies are located in the same directory. Hammond discloses such a scenario (see, for example, column 5, lines 30-32, "often, the correct DLLs are located in the application's own directory," or column 5, lines 55-57, "the 'shared' directory may conveniently hold particular versions of DLLs that are used by many applications").

Note that 35 U.S.C. 132(a) states that no amendment shall introduce new matter into the disclosure of the invention.

4. It is noted that Applicant reiterates other arguments that were addressed and found not persuasive in previous Office actions.

Furthermore, Applicant now states, "A version number is indicative of a point-in-time sequential build, e.g., version 2.0 is newer and intended to be better than version 1.0" (remarks, page 22, top).

The examiner does not disagree with this statement. However, none of the claims recites the term "version number," much less any limitation that the version number is to indicate a point-in-time sequential build.

Again, although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Response to Amendment

5. The objection to the specification is withdrawn in view of Applicant's amendment.

6. The rejection of claims 15, 31-41 and 49 under 35 U.S.C. 101 is withdrawn in view of Applicant's amendment to limit the claims to statutory subject matter per the *Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility* (1300 OG 142).

Terminal Disclaimer

7. The terminal disclaimer filed on June 12, 2006 disclaiming the terminal portion of any patent granted on this application that would extend beyond the expiration date of U.S. Patent No. 6,871,344 has been reviewed and is accepted. The terminal disclaimer has been recorded.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

9. Claims 1-17, 19-22, 25-31, 42, 43, 45-47 and 49 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 5,974,470 to Hammond (art of record, "Hammond").

With respect to claim 1 (currently amended), Hammond discloses a computer-implemented method (see, for example, the abstract), comprising:

(a) receiving a request from executable code to load an assembly, the request not including assembly version data (see, for example, column 5, line 58 to column 6, line 12, which shows receiving a request from an executable application to load a module or assembly), including when the assembly is among a plurality of assemblies having at least some components located in a same directory (see, for example, column 5, lines 30-34 and 54-57, which shows that a plurality of DLLs or assemblies are located in a same directory);

(b) building an activation context that distinguishes between versions of assemblies based on an actual version, the activation context associated with the executable code in response to the request to load an assembly (see, for example, column 8, lines 23-58, which shows building databases or an activation context associated with the application to load the DLL or assembly under an alias and distinguish among versions of the DLL or assembly based on a version key, and column 7, lines 59-64, which shows that the version key specifies the particular or actual version of the DLL or assembly);

(c) consulting information in a manifest associated with the executable code and stored separate from the executable code to determine a particular version of the assembly in response to the building of the activation context (see, for example, column 7, line 51 to column 8, line 5, which shows consulting rules or information associated with the application to determine a particular version of the DLL or assembly, and column 5, lines 7-18, which shows that the rules or information is stored in a separate configuration file or manifest); and

(d) providing the particular version of the assembly for use by the executable code (see, for example, column 5, lines 27-30, which shows providing the correct version of the DLL or assembly).

With respect to claim 2 (original), the rejection of claim 1 is incorporated, and Hammond further discloses the limitation wherein the request corresponds to a request to load a privatized assembly (see, for example, column 5, lines 30-34, which shows that the DLL or assembly may be located in the application's directory, i.e. the requested assembly may be privatized).

With respect to claim 3 (original), the rejection of claim 1 is incorporated, and Hammond further discloses the limitation wherein the request corresponds to a request to load a shared assembly (see, for example, column 5, lines 52-57, which shows that the DLL or assembly may be shared by many applications).

With respect to claim 4 (original), the rejection of claim 3 is incorporated, and Hammond further discloses the limitation wherein the shared assembly is maintained in an assembly cache (see, for example, column 5, lines 52-57, which shows that the shared DLL or assembly is located in a designated directory, i.e. in an assembly cache).

With respect to claim 5 (original), the rejection of claim 1 is incorporated, and Hammond further discloses the limitation wherein consulting information associated with the executable code to determine a particular version of the assembly includes searching for a mapping from a version independent name provided by the executable code to a version specific assembly (see, for example, column 7, line 51 to column 8, line 5, which shows searching for a rule or mapping from the requested DLL name to a specific version of the DLL or assembly).

With respect to claim 6 (original), the rejection of claim 5 is incorporated, and Hammond further discloses the limitation wherein no mapping from the version independent name to a

version specific assembly is present, and wherein providing the particular version of the assembly for use by the executable code comprises providing a default version (see, for example, column 6, lines 57-59, which shows providing the default module or assembly when no rules or mappings are present).

With respect to claim 7 (original), the rejection of claim 1 is incorporated, and Hammond further discloses the limitation wherein providing the particular version of the assembly comprises accessing a file corresponding to the assembly and loading the assembly into memory from the file (see, for example, column 5, lines 7-18, which shows that the rules for providing the particular version of a DLL or assembly for loading into memory are accessed from a file).

With respect to claim 8 (original), the rejection of claim 1 is incorporated, and Hammond further discloses the limitation wherein the information associated with the executable code includes a mapping between a version independent name provided by the executable code and a version specific file system path and filename of the particular version of the assembly, and wherein providing the particular version of the assembly comprises returning the path and filename to an assembly loading mechanism (see, for example, column 5, line 58 to column 6, line 12, which shows returning the fully qualified path of the particular version of the module or assembly to the loading routine).

With respect to claim 9 (original), the rejection of claim 8 is incorporated, and Hammond further discloses the limitation wherein the executable code is stored as an application executable file in a folder, and wherein the version of the assembly is stored as another file in the same

folder (see, for example, column 5, lines 30-34, which shows that the application is stored as an executable file in a directory and the DLL or assembly may be located in the same directory).

With respect to claim 10 (original), the rejection of claim 8 is incorporated, and Hammond further discloses the limitation wherein the filename corresponds to a file in an assembly cache (see, for example, column 6, lines 40-54, which shows that the fully qualified path corresponds to a file in a shared directory, i.e. in an assembly cache).

With respect to claim 11 (original), the rejection of claim 1 is incorporated, and Hammond further discloses the limitation wherein the information associated with the executable code is derived from application manifest (see, for example, column 5, lines 7-18, which shows that the information is stored in a configuration file associated with the application, i.e. in an application manifest).

With respect to claim 12 (original), the rejection of claim 11 is incorporated, and Hammond further discloses the limitation wherein the information associated with the executable code is further derived from at least one assembly manifest (see, for example, column 5, lines 7-18, which shows that the information is stored in configuration files associated with the DLLs or assemblies, i.e. in assembly manifests).

With respect to claim 13 (original), the rejection of claim 1 is incorporated, and Hammond further discloses the limitation wherein the information associated with the executable code is constructed during a pre-execution initialization phase (see, for example, column 5, lines

19-25, which shows that the information may be constructed at any designated time, such as upon installation of the application, i.e. during an initialization phase prior to execution).

With respect to claim 14 (original), the rejection of claim 1 is incorporated, and Hammond further discloses the limitation wherein the information associated with the executable code is persisted into a non-volatile memory (see, for example, column 5, lines 7-18, which shows storing or persisting the information in a file, i.e. in non-volatile memory).

With respect to claim 15 (currently amended), the rejection of claim 1 is incorporated, and Hammond further discloses a computer-readable storage medium having computer-executable instructions for performing the recited method (see, for example, column 5, lines 33-49, which shows applying software patches, i.e. computer-executable instructions, to an operating system inherently stored on a computer-readable storage medium).

With respect to claim 16 (currently amended), Hammond discloses a computer-implemented method (see, for example, the abstract), comprising:

(a) building an activation context that distinguishes between versions of assemblies based on an actual version, the activation context associated with executable code, the activation context identifying dependency information (see, for example, column 8, lines 23-58, which shows building databases or an activation context associated with the application to load the DLL or assembly under an alias and distinguish among versions of the DLL or assembly based on a version key, and column 7, lines 59-64, which shows that the version key specifies the particular or actual version of the DLL or assembly, and see, for example, column 9, lines 25-40, which shows that the activation context identifies dependency information);

(b) interpreting the dependency information associated with the executable code, the dependency information identifying at least one particular version of an assembly (see, for example, column 7, line 51 to column 8, line 5, which shows interpreting dependency information associated with the application to determine a particular version of the DLL or assembly) including when the assembly is among a plurality of assemblies having at least some components located in a same directory (see, for example, column 5, lines 30-34 and 54-57, which shows that a plurality of DLLs or assemblies are located in a same directory); and

(c) associating with the executable code at least one mapping based on the dependency information, each mapping relating a version independent assembly name that the executable code may provide to a version specific assembly identified in the dependency information (see, for example, column 7, line 51 to column 8, line 5, which shows associating with the application a rule or mapping from the requested DLL name to a specific version of the DLL or assembly).

With respect to claim 17 (original), the rejection of claim 16 is incorporated, and Hammond further discloses the limitation wherein the dependency information is provided in an application manifest associated with the executable code (see, for example, column 5, lines 7-18, which shows that the information is provided in a configuration file associated with the application, i.e. in an application manifest).

With respect to claim 19 (original), the rejection of claim 16 is incorporated, and Hammond further discloses the limitation wherein at least one mapping maps a version independent name to an assembly stored in a common folder with an application executable file that corresponds to the executable code (see, for example, column 5, lines 30-34, which shows

that the application is an executable file in a directory and the DLL or assembly may be commonly located in the same directory).

With respect to claim 20 (original), the rejection of claim 16 is incorporated, and Hammond further discloses the limitation wherein at least one mapping maps a version independent name to a shared assembly in an assembly cache (see, for example, column 6, lines 40-54, which shows mapping the requested name to a DLL or assembly located in a shared directory, i.e. in an assembly cache).

With respect to claim 21 (original), the rejection of claim 16 is incorporated, and Hammond further discloses the limitation wherein the dependency information provided by the executable code corresponds to an assembly having an assembly manifest associated therewith, and further comprising, interpreting the assembly manifest (see, for example, column 5, lines 7-18, which shows that the information is stored in configuration files associated with the DLLs or assemblies, i.e. in assembly manifests).

With respect to claim 22 (original), the rejection of claim 21 is incorporated, and Hammond further discloses the limitation wherein the assembly manifest specifies that a particular version of an assembly be replaced with another version of that assembly (see, for example, column 8, lines 6-22, which shows replacing a version of a DLL or assembly with another specified version).

With respect to claim 25 (original), the rejection of claim 16 is incorporated, and Hammond further discloses the limitation wherein the at least one mapping is maintained in an

activation context, and further comprising, persisting the activation context (see, for example, column 8, lines 23-58, which shows alias mappings maintained in a database, i.e. in an activation context; also see, for example, column 9, lines 25-40, which shows persisting the activation context in the form of an alias file).

With respect to claim 26 (original), the rejection of claim 25 is incorporated, and Hammond further discloses the limitation wherein associating with the executable code the at least one mapping comprises retrieving a persisted activation context (see, for example, column 9, lines 25-32, which shows retrieving a persisted alias file or activation context).

With respect to claim 27 (original), the rejection of claim 25 is incorporated, and Hammond further discloses the limitation wherein associating with the executable code the at least one mapping comprises constructing a new activation context (see, for example, column 9, lines 32-40, which shows constructing a new alias file or activation context).

With respect to claim 28 (original), the rejection of claim 27 is incorporated, and Hammond further discloses the limitation wherein the new activation context is constructed upon determining that an activation context does not exist (see, for example, column 9, lines 32-40, which shows constructing the new alias file or activation context when one does not exist).

With respect to claim 29 (original), the rejection of claim 27 is incorporated, and Hammond further discloses the limitation wherein the new activation context is constructed upon determining that an existing activation may not be not coherent with current policy (see, for example, column 9, lines 1-40, which shows determining whether a loaded module, i.e. an

existing activation, is the correct version, i.e. is coherent with current policy, and then constructing a new alias file or activation context).

With respect to claim 30 (original), the rejection of claim 16 is incorporated, and Hammond further discloses running the executable code, receiving a request from the executable code to load an assembly, the request including data corresponding to a version independent name of the assembly and providing a particular version of the assembly for use by the executable code based on a mapping therefor (see, for example, column 5, line 58 to column 6, line 12, which shows receiving a request from a running executable application to load a module or assembly and providing the correct version; also see, for example, column 7, line 51 to column 8, line 5, which shows determining the correct version of the DLL or assembly based on rules or mappings).

With respect to claim 31 (currently amended), the rejection of claim 16 is incorporated, and Hammond further discloses a computer-readable storage medium having computer-executable instructions for performing the recited method (see, for example, column 5, lines 33-49, which shows applying software patches, i.e. computer-executable instructions, to an operating system inherently stored on a computer-readable storage medium).

With respect to claim 42 (currently amended), Hammond discloses a system in a computing environment (see, for example, the abstract), comprising:

(a) an initialization mechanism configured to interpret dependency data associated with executable code, the dependency data corresponding to at least one assembly version on which the executable code depends (see, for example, column 5, lines 27-30, which shows an

initialization mechanism, and column 7, line 51 to column 8, line 5, which shows interpreting dependency information associated with an executable application to determine a particular version of a DLL or assembly needed by the application), each assembly version corresponding to an assembly having version information associated therewith (see, for example, column 7, lines 31-35, which shows version information associated with the module or assembly) and contained in a directory structure among a plurality of assemblies (see, for example, column 5, lines 30-34 and 54-57, which shows that a plurality of DLLs or assemblies are located in a same directory);

(b) an activation context that distinguishes between versions of assemblies based on an actual version, the activation context associated with the executable code and constructed by the initialization mechanism based on the dependency data, the activation context relating at least one version independent assembly identifier provided by the executable code to a version specific assembly (see, for example, column 8, lines 23-58, which shows databases or an activation context associated with the application that relates the requested DLL name to an alias for a specific version of the DLL or assembly and distinguishes among versions of the DLL or assembly based on a version key, and column 7, lines 59-64, which shows that the version key specifies the particular or actual version of the DLL or assembly); and

(c) a version matching mechanism configured to access the activation context to relate a version independent request from the executable code to a version specific assembly (see, for example, column 7, line 51 to column 8, line 5, which shows matching the requested DLL name with a specific version of the DLL or assembly).

With respect to claim 43 (original), the rejection of claim 42 is incorporated, and Hammond further discloses the limitation wherein the dependency data is included in executable code manifest (see, for example, column 5, lines 7-18, which shows that the dependency data is stored in a configuration file associated with the application, i.e. in an executable code manifest).

With respect to claim 45 (original), the rejection of claim 42 is incorporated, and Hammond further discloses the limitation wherein the initialization mechanism persists the activation context (see, for example, column 9, lines 25-40, which shows persisting the activation context in the form of an alias file).

With respect to claim 46 (original), the rejection of claim 42 is incorporated, and Hammond further discloses an assembly loading mechanism configured to communicate with the executable code and the version matching mechanism to load the version specific assembly upon a request by the executable code to load a requested assembly, wherein the request does not include version specific data (see, for example, column 5, line 58 to column 6, line 12, which shows loading the correct version of the module or assembly requested by the executable application).

With respect to claim 47 (original), the rejection of claim 46 is incorporated, and Hammond further discloses the limitation wherein the assembly loading mechanism loads the version specific assembly from an assembly cache (see, for example, column 5, lines 52-57, which shows that the DLL or assembly is located in a shared directory, i.e. in an assembly cache).

With respect to claim 49 (original), the rejection of claim 42 is incorporated, and Hammond further discloses a computer-readable medium having computer-executable modules configured to implement the recited system (see, for example, column 5, lines 33-49, which shows applying software patches, i.e. computer-executable modules, to an operating system inherently stored on a computer-readable medium).

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 18, 24 and 44 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hammond, as applied to claims 17, 16 and 42 above, respectively.

With respect to claim 18 (original), the rejection of claim 17 is incorporated, and although Hammond discloses a common directory storing both the DLLs or assemblies and the executable file of the application (see, for example, column 5, lines 30-34), Hammond is silent as to the location of the configuration file or application manifest (see, for example, column 5, lines 7-18). Accordingly, Hammond does not expressly disclose the limitation wherein the application manifest is associated with the executable code by being stored in a common folder with an application executable file that corresponds to the executable code.

However, is well known in the art that configuration files or manifests and other application files may be stored in the same directory as the executable file. The advantage of such an arrangement is that the operating system may find the files using the predetermined search order (see, for example, Hammond, column 5, lines 30-34).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to store the configuration file or application manifest taught by Hammond in a common folder along with the application executable file that corresponds to the executable code, so that the operating system may successfully find the file.

With respect to claim 24 (original), the rejection of claim 16 is incorporated, and although Hammond discloses interpreting dependency information (see, for example, column 7, line 51 to column 8, line 5) in response to a request from a running application to load a module or assembly (see, for example, column 5, line 58 to column 6, line 12), Hammond does not expressly disclose the limitation wherein the dependency information is interpreted in response to receiving a request to execute the executable code.

However, Hammond further discloses that the modules or assemblies are dynamic-link libraries, which are loaded and linked at run time, i.e. when the executable code is executed (see, for example, column 1, lines 23-32). It is well known in the art that DLLs or other assemblies may be required immediately upon execution of an application.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform the interpretation shown by Hammond in response to receiving a request to execute the executable code, because that request may require a particular version of a DLL or assembly.

With respect to claim 44 (original), the rejection of claim 42 is incorporated, and although Hammond discloses storing the dependency data in a configuration file (see, for example, column 5, lines 7-18), Hammond does not expressly disclose the limitation wherein the dependency data is included in an XML file.

However, it is well known in the art that XML is a flexible, standard markup language for structuring and organizing data.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to format the configuration file taught by Hammond as an XML file, for the purpose of structuring the dependency data using a standard language.

12. Claims 23 and 48 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hammond, as applied to claims 21 and 42 above, respectively, in view of U.S. Patent No. 6,185,734 to Saboff et al. (art of record, "Saboff").

With respect to claim 23 (original), the rejection of claim 21 is incorporated, and although Hammond discloses dependency information for an executable application (see, for example, column 7, line 51 to column 8, line 5), Hammond does not expressly disclose the limitation wherein the assembly manifest specifies at least one particular version of another assembly on which the assembly having an assembly manifest is dependent.

However, Saboff discloses a registry structure that serves as an assembly manifest for managing versions of software components (see, for example, the abstract, and FIGS. 5 and 6), wherein the manifest specifies the assemblies upon which a first assembly is dependent (see, for

example, column 7, line 61 to column 8, line 14). The dependency information specifies the files that are required by a library or assembly (see, for example, column 7, lines 61-64).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the assembly manifest of Hammond with the additional dependency information taught by Saboff, for the purpose of specifying the files required by a particular version of an assembly.

With respect to claim 48 (original), the rejection of claim 42 is incorporated, and although Hammond discloses dependency information for an executable application (see, for example, column 7, line 51 to column 8, line 5) and adding such information to an activation context (see, for example, column 8, lines 23-58), Hammond does not expressly disclose the limitation wherein the dependency data identifies an assembly that has assembly dependency data associated therewith, the assembly dependency data corresponding to at least one other assembly version on which the assembly depends.

However, Saboff discloses a registry structure that serves as an assembly manifest for managing versions of software components (see, for example, the title and abstract, and FIGS. 5 and 6), wherein the manifest specifies the assemblies upon which a first assembly is dependent (see, for example, column 7, line 61 to column 8, line 14). The dependency information specifies the files that are required by a library or assembly (see, for example, column 7, lines 61-64).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the assembly manifest of Hammond with the additional dependency

information taught by Saboff, for the purpose of specifying the files required by a particular version of an assembly.

13. Claims 32-41 are rejected under 35 U.S.C. 103(a) as being unpatentable over Saboff in view of Hammond.

With respect to claim 32 (currently amended), Saboff discloses a computer-readable storage medium having stored thereon a data structure (see, for example, the abstract, and FIGS. 5 and 6), comprising:

(a) a first data store operable to store a first set of data comprising a name of an assembly (see, for example, service 509 in FIG. 6, and column 9, lines 11-18).

Saboff does not expressly disclose that the first data store is operable to store a first set of data comprising a name of an assembly including when the assembly is among a plurality of assemblies having at least some components located in a same directory.

However, it is well known in the art that a plurality of assemblies may have at least some components located in a same directory. In fact, Hammond expressly discloses a plurality of DLLs or assemblies having at least some components located in a same directory (see, for example, column 5, lines 30-34 and 54-57).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made that the first data store in Saboff is operable to store a first set of data comprising a name of an assembly including when the assembly is among a plurality of assemblies having at least some components located in a same directory.

Saboff in view of Hammond further discloses:

(b) a second data store operable to store a second set of data comprising a version of the assembly (see, for example, version 513 in FIG. 6, and column 9, lines 53-55);

(c) a third data store operable to store a third set of data comprising at least one item of the assembly (see, for example, state 510 and type 511 in FIG. 6, and column 9, lines 19-39); and

(d) a fourth data store operable to store a fourth set of data comprising binding path data to each item in the third set of data (see, for example, path 507 in FIG. 6, and column 9, lines 5-7);

(e) wherein each data store is operable to provide information to an activation context that distinguishes between versions of assemblies based on an actual version when executable code is executed (see, for example, column 5, lines 22-44, which shows providing information to the activation context of an application when it is executed to distinguish among versions of the library service or assembly, and column 9, lines 53-55, which shows that the versions are actual version numbers).

With respect to claim 33 (original), the rejection of claim 32 is incorporated, and Saboff further discloses the limitation wherein the binding path data comprises a location of a dynamic link library (see, for example, column 9, lines 53-55, which shows the location of a file, and lines 33-39, which shows that the file may correspond to a library, i.e. to a dynamic-link library).

With respect to claim 34 (original), the rejection of claim 32 is incorporated, and Saboff further discloses the limitation wherein the binding path data comprises an object class identifier (see, for example, identifier 506 in FIG. 6, and column 9, lines 1-4, which shows a unique identifier that may serve as an object class identifier).

With respect to claim 35 (original), the rejection of claim 32 is incorporated, and Saboff further discloses the limitation wherein the binding path data comprises a programmatic identifier (see, for example, identifier 506 in FIG. 6, and column 9, lines 1-4, which shows a unique identifier that may serve as a programmatic identifier).

With respect to claim 36 (original), the rejection of claim 32 is incorporated, and Saboff further discloses a fifth set of data comprising data corresponding to at least one dependency on an assembly (see, for example, dependencies 504 in FIG. 6, and column 8, lines 57-59).

With respect to claim 37 (original), the rejection of claim 32 is incorporated, and Saboff further discloses a fifth set of data comprising data corresponding to a Windows® class (see, for example, extensions 505 in FIG. 6, and column 8, lines 60-67, which shows data corresponding to the capabilities provided by a service, for example such as a Windows® class).

With respect to claim 38 (original), the rejection of claim 32 is incorporated, and Saboff further discloses a fifth set of data comprising data corresponding to a global name (see, for example, column 9, lines 11-18, which shows that the service name is an abstract or global name).

With respect to claim 39 (currently amended), Saboff discloses a computer-readable storage medium having stored thereon a data structure (see, for example, the abstract, and FIGS. 5 and 6), comprising:

(a) a first data store operable to store a first set of data comprising a version independent name of an assembly (see, for example, service 509 in FIG. 6, and column 9, lines 11-18).

Saboff does not expressly disclose that the first data store is operable to store a first set of data comprising a version independent name of an assembly including when the assembly is among a plurality of assemblies having at least some components located in a same directory.

However, it is well known in the art that a plurality of assemblies may have at least some components located in a same directory. In fact, Hammond expressly discloses a plurality of DLLs or assemblies having at least some components located in a same directory (see, for example, column 5, lines 30-34 and 54-57).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made that the first data store in Saboff is operable to store a first set of data comprising a version independent name of an assembly including when the assembly is among a plurality of assemblies having at least some components located in a same directory.

Saboff in view of Hammond further discloses:

(b) a second data store operable to store a second set of data comprising a filename path to a specific version of the assembly (see, for example, path 507 in FIG. 6, and column 9, lines 5-7);

(c) wherein the second set of data is associated with the first set of data such that a reference to the version independent name in the first set of data is mapped to the specific version of the assembly via the second set of data (see, for example, service 509 and path 507 in FIG. 6, and column 9, lines 5-7 and 11-18); and

(d) wherein each data store is operable to provide information to an activation context that distinguishes between versions of assemblies based on an actual version when executable code is executed (see, for example, column 5, lines 22-44, which shows providing information to

the activation context of an application when it is executed to distinguish among versions of the library service or assembly, and column 9, lines 53-55, which shows that the versions are actual version numbers).

With respect to claim 40 (original), the rejection of claim 39 is incorporated, and Saboff further discloses a third set of data comprising a version independent object class name (see, for example, column 9, lines 11-18, which shows an abstract name that may serve as an object class name), a fourth set of data comprising an assembly name corresponding to a file that contains an object class that corresponds to the object class name in the third set of data (see, for example, column 9, lines 5-7, which shows a path to a file), and a fifth set of data comprising a version specific name that corresponds to the third set of data (see, for example, version 513 in FIG. 6, and column 9, lines 53-55).

With respect to claim 41 (currently amended), Saboff discloses a computer-readable storage medium having stored thereon a data structure (see, for example, the abstract, and FIGS. 5 and 6), comprising:

(a) a first data store operable to store a first set of data comprising a version independent object class name (see, for example, service 509 in FIG. 6, and column 9, lines 11-18, which shows an abstract name that may serve as an object class name);

(b) a second data store operable to store a second set of data comprising an assembly name corresponding to a file that contains an object class that corresponds to the object class name in the first set of data (see, for example, path 507 in FIG. 6, and column 9, lines 5-7, which shows a path to a file).

Saboff does not expressly disclose that the second data store is operable to store a second set of data comprising an assembly name corresponding to a file that contains an object class that corresponds to the object class, including when the file is among a plurality of files having objects located in a same directory.

However, it is well known in the art that a plurality of files may have objects located in a same directory. In fact, Hammond expressly discloses a plurality of DLLs or files having objects located in a same directory (see, for example, column 5, lines 30-34 and 54-57).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made that the second data store in Saboff is operable to store a second set of data comprising an assembly name corresponding to a file that contains an object class that corresponds to the object class, including when the file is among a plurality of files having objects located in a same directory.

Saboff in view of Hammond further discloses:

(c) a third data store operable to store a third set of data comprising a version specific name that corresponds to the first set of data such that a reference to the version independent name in the first set of data is mapped to the specific version of the object class (see, for example, version 513 in FIG. 6, and column 9, lines 53-55);

(d) wherein each data store is operable to provide information to an activation context that distinguishes between versions of assemblies based on an actual version when executable code is executed (see, for example, column 5, lines 22-44, which shows providing information to the activation context of an application when it is executed to distinguish among versions of the

library service.or assembly, and column 9, lines 53-55, which shows that the versions are actual version numbers).

Conclusion

14. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

TUAN DAM
SUPERVISORY PATENT EXAMINER